April 1991
Vol. 5, No. 4

The Ultimate File System: The Role of Religion in the Enterprise-Wide Network

by Carl Malamud

[Editor's Note: In our November 1990 issue (Volume 4, No. 11) we looked at the question of FTAM, FTP, and NFS as three candidates for the enterprise-wide file access mechanism. Eric Fleischman argued that NFS and FTP were poorly suited for this role and nominated FTAM as the enterprise-wide solution. In this article, Carl Malamud argues that comparing NFS and FTAM is like comparing apples and oranges].

Introduction

It has recently been in vogue to compare FTAM to NFS and FTP. Comparing any protocol to FTP is kind of like beating up an old lady instead of helping her across the street. Let us, for the time being, leave FTP standing on the corner waiting for an escort, and look instead at FTAM and NFS.

Mangoes or Orangutans?

Comparing NFS and FTAM is like comparing mangoes to orangutans. If you pick a high enough metaphysical viewpoint, they both serve the same function—both are carbon-based life forms. With a given set of criteria, one can easily compare the two and come up with a clear winner: "The mango is more portable and is thus preferable to the orangutan."

If you take a lower perspective, differences between the mango and the orangutan start to show up. The mango certainly tastes better, but you can't take a group of kids to the zoo to watch a mango and expect them to stay entertained.

It is easy to fall into the mango trap when looking at emerging standards like FTAM. When we see an international standard emerging, it is tempting to say that it will solve all problems and that it should be used to the exclusion of other protocols. After all, we don't want duplication of effort.

The question is the universe used for the comparison. Analyzing enterprise-wide file access mechanisms to pick a winner requires some limitation of scope. Of course, one could compare Telnet to FTAM and decide that FTAM was better: even die-hard FTAM fanatics would consider such an analysis silly.

In the area more traditionally known as file access mechanisms, one can pick a small subset, such as FTP, FTAM, and the Network File System. One could extend this analysis to include things as the Carnegie Mellon's *Andrew File System* (AFS), DEC's *Data Access Protocol,* Novell's *NetWare Core Protocols.* One could (but hopefully would not) even go so far as to consider thing's like RJE over Bisync or custom COBOL programs over X.25 as candidates.

The Ultimate File System

To ease this analysis, I would like to contribute another imaginary candidate which I'll call the *Ultimate File System* (UFS). UFS is clearly better than all other candidates: it is fast, efficient, powerful. It slices, it dices, and can be implemented quickly and easily. It has an installed base only slightly smaller than FTAM.

Since UFS is clearly better, one could do a blow by blow comparison to FTAM and NFS. There is no doubt that, given any set of criteria, one can find holes in each of them. The basic problem with choosing a single enterprise-wide solution between candidates like UFS, NFS, and FTAM is that they really do different things.

The Ultimate File System *(continued)*

If we look at NFS, we will see that there is no concept of document types. Is this a problem? Not if we understand that the role of NFS is to leave that interpretation to the client. Do you really want the operating system for your diskless VAXstation encoded so a Sun could read it? What would it do with it?

Why make the user of the data interpret the data? It certainly reduces server load. It promotes the ability of a client to store arbitrary data on arbitrary servers. Finally, the client can access raw blocks at speeds much better than the structured access: ideal for things like diskless nodes and other applications where the client is accessing programs and not data.

This is not meant to imply that you don't have independence of data on a network in NFS. The *External Data Representation* (XDR) allows a user to take data structures and encode them for network presentation. This RPC function of encoding data is used by many applications, such as Frame desktop publishing or the Legato network backup utility.

Should NFS be structuring files? Binary data doesn't have much interchange across machine architectures. Either the application knows how to use it or it doesn't. If you need interchange of arbitrary data structures, you're looking at an RPC question, not a distributed file system.

Let's stop and look at this point again: NFS and FTAM do different things so they have a different approach to data translation. NFS is a file server: FTAM is a structured, record-oriented virtual file system.

Stateful and Stateless Protocols

There are some other differences between the two. FTAM uses a connection-oriented, stateful approach, whereas NFS is stateless and uses (at least in most implementations) the UDP transport layer in TCP/IP.

Again, the protocols do different things. FTAM is inherently stateful and has a built in lock facility. NFS allows the locking mechanism to be an independent service. This doesn't mean that an operating system is going to allow multiple users to walk all over data: just because the NFS protocol is stateless doesn't mean that the server doesn't keep state information through a lock manager, a duplicate request cache, or application semaphores.

Note that making NFS stateless simplifies it's role as a file service, making crash recovery and other operations significantly easier. This doesn't mean that NFS can't work over a connection-oriented transport service: the Reno tape for the new Berkeley UNIX features a TCP-based implementation of NFS that works quite well in a wide-area environment.

Can NFS and FTAM coexist in a single network? I've seen lots of DEC systems that use the *Record Management Services* (RMS) to tie together NFS, FTAM, DEC's Data Access Protocols, VAX Clusters, and DEC's Distributed File System into a single coherent view of data in a wide variety of environments. The local file access mechanism, in this case the Record Management Services is responsible for masking the different protocols from the client application.

Scaling Protocols

A final comment on NFS versus FTAM is the often-mentioned bromism that "NFS does not scale well." It is true that a UDP-based transport connection for large files does not work well in a wide-area environment: hence the Reno distribution with the TCP transport.

In a local or enterprise-wide network, I would argue that NFS has certainly scaled quite well. Technologies like the *automounter* and the *NIS name service* make resource discovery and automatic mounting simple. Dedicated file servers from companies like Auspex and Epoch allow terabytes of NFS-accessible data to be available to clients.

Lists of Three

It is sometimes tempting to group all applications on all networks into three categories: data access, mail, and virtual terminals. Once this taxonomy is heard enough times, it starts to make sense. Every protocol is put into one of the three categories. The next step is often to find the single best protocol within the three categories. The result is a list of the three standard protocols that make up a global solution to all our problems.

This approach is simplistic at best and can be quite destructive. Computer networks do many different things for many different populations of users. Picking a simple protocol is nice, but using it to the exclusion of complementary services just reduces the user's capability to accomplish useful work.

FTAM versus COBOL

Take FTAM. Making that the enterprise-wide solution ignores distributed file systems, distributed databases, transactions processing, and a wide variety of other data access paradigms. FTAM resembles COBOL in many respects. Given a universe of COBOL, RPG II and BASIC, standardizing on one or the other helps. But specifying COBOL over complementary languages just ossifies the organization.

Let me give you an example. I used to work in the research division of the Board of Governors of the Federal Reserve System. We were trying to provide an alternative to MVS/TSO for econometric analysis and needed to hire some programmers to port our applications to UNIX. Unfortunately, MIS had decided that the standard language skill needed to work at the Fed was a facility with COBOL. Needless to say, econometric forecasting with COBOL is a little tough. Because MIS had decided COBOL was our enterprise-wide standard, Personnel wouldn't let us advertise for FORTRAN or **C** skills. Instead, we had to put an ad out looking for COBOL programmers and hope that somebody listed FORTRAN or **C** on their resume.

Enterprise-wide solutions should not attempt to provide a lowest common denominator or form a rigid model. Simplistic solutions to difficult problems don't help anybody. This doesn't mean we don't need standards. Standards are great for interoperability. Standards should be a means to interoperability, however, not a corporate religion. A given enterprise may wish to declare a corporate religion. FTAM may be an article of faith within a corporation, but religions are often difficult for the populace to live up to. Ideals to contemplate, perhaps, but in the real world truth and beauty often lose out to whatever seems best at the time.

UFS+ ?

Let's return to the question of the Ultimate File System. I'm not going to recommend that you discard FTAM and NFS in favor of UFS, even though UFS is clearly better. Instead, I would begin implementing UFS on systems in tandem with FTAM and NFS. I might even make a UFS to FTAM gateway system to ease transition. As more and more of my users begin to use UFS, I might even make it one of several required applications on any host on my network. But I certainly wouldn't say this was my network-wide, enterprise-wide solution. After all, UFS+ will be available soon afterwards.

CARL MALAMUD considers himself a secular humanist at layer 10 of the ISO Reference Model.